

UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF CALIFORNIA

THOUGHT, INC.,
Plaintiff,

v.

ORACLE CORPORATION, et al.,
Defendants.

Case No. [12-cv-05601-WHO](#)

CLAIM CONSTRUCTION ORDER

Re: Dkt. No. 70

BACKGROUND

The parties ask the Court to construe 19 terms from the seven patents asserted in this case. As explained by plaintiff Thought's expert, Hosagrahar Jagadish, Ph.D., the patents in suit are aimed at facilitating the transfer of information between two popular schemes for organizing information in a computer: the relational model and the object model. Declaration of Hosagrahar Jagadish in Response (Jagadish Resp. Decl.; Docket No. 76-1) ¶ 11. The relational model is frequently used in database systems, while the object model is used in most programming languages and software applications. *Id.* Generally speaking, the technology at issue teach methods to allow ordinary users to make object (software) applications and relational databases work together, despite the differences in the ways software applications and database systems organize data.

More specifically, the '197 Patent (Patent Number 5,857,197, issued January 5, 1999), "System and Method for Accessing Data Stores as Objects," teaches the use of an adapter abstraction layer using two adapters (interfaces). The novelty, according to Thought, is the fact that the two adapters or interfaces are used at runtime and one specializes in the object model and the other specializes in the relational database model. Thought argues that the '197 Patent teaches

1 the innovative “basic persistence architecture” of the method (which was incorporated into
2 Thought’s “CocoBase” product), and advanced features related to development, modeling, and
3 caching for CocoBase were claimed in the following patents.

4 The ‘600 Patent (Patent Number 7,103,600, issued September 5, 2006) is titled
5 “Displayable Presentation Page and SQL Searchable Relational Data Source Implementation of a
6 System, Method and Software for Creating or Maintaining Distributed Transparent Persistence of
7 Complex Data Objects and Their Data Relationship.” The ‘862 Patent (Patent Number 7,167,62,
8 issued January 23, 2007) is titled, “Session Bean Implementation of a System, Method and
9 Software for Creating or Maintaining Distributed Transparent Persistence of Complex Data
10 Objects and Their Data Relationships.” These two patents teach how to maintain persistence (to
11 save or to store) for objects and their relationships.

12 The ‘481 Patent (7,043,481, issued May 9, 2006), “System, Method and Software for
13 Creating, Maintaining, Navigating or Manipulating Complex Data Objects and Their Data
14 Relationships,” discloses a Complex Data Object Graph (CDOG) model to capture the “web” of
15 relationships between objects, also called a “graph.” The patent teaches how to represent, manage,
16 an access the graph effectively.

17 The ‘956 Patent (6,999,956, issued February 14, 2006), “Dynamic Object-Driven Database
18 Manipulation and Mapping System,” teaches how to map data between the object and relational
19 programs using metadata in a library. Jagadish Resp. Decl. ¶ 14.

20 The ‘730 Patent (7,149,730, issued December 12, 2006), “Dynamic Class Inheritance and
21 Distributed Caching with Object Relational Mapping and Cartesian Model Support in a Database
22 Manipulation and Mapping System,” teaches how to work with multiple data stores that contain
23 information organized differently.

24 The ‘912 Patent (6,985,912, issued January 10, 2006), “Dynamic Object-Driven Database
25 Manipulation and Mapping System Having a Simple Global Interface and an Optional Multiple
26 User Need Only Caching System With Disable and Notify Features,” addresses the problem of
27 user manipulation of stored data, teaching how to keep copies of data synchronized.

28 The 19 claim terms in dispute on this motion can be broken down into three categories,

discussed below.

LEGAL STANDARD

Claim construction is a matter of law. *See Markman v. Westview Instruments, Inc.*, 517 U.S. 370, 372 (1996); *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996). Terms contained in claims are “generally given their ordinary and customary meaning.” *Vitronics*, 90 F.3d at 1582. In determining the proper construction of a claim, a court begins with the intrinsic evidence of record, consisting of the claim language, the patent specification, and, if in evidence, the prosecution history. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1313 (Fed. Cir. 2005); *see also Vitronics*, 90 F.3d at 1582. “A claim term used in multiple claims should be construed consistently” *Inverness Med. Switzerland GmbH v. Princeton Biomeditech Corp.*, 309 F.3d 1365, 1371 (Fed. Cir. 2002).

“The appropriate starting point [] is always with the language of the asserted claim itself.” *Comark Commc’ns, Inc. v. Harris Corp.*, 156 F.3d 1182, 1186 (Fed. Cir. 1998). “[T]he ordinary and customary meaning of a claim term is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention, i.e., as of the effective filing date of the patent application.” *Phillips*, 415 F.3d at 1312. “There are only two exceptions to this general rule: 1) when a patentee sets out a definition and acts as his own lexicographer, or 2) when the patentee disavows the full scope of a claim term either in the specification or during prosecution.” *Thorner v. Sony Computer Entm’t Am. LLC*, 669 F.3d 1362, 1365 (Fed. Cir. 2012).

“Importantly, the person of ordinary skill in the art is deemed to read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification.” *Phillips*, 415 F.3d at 1313. “Claims speak to those skilled in the art,” but “[w]hen the meaning of words in a claim is in dispute, the specification and prosecution history can provide relevant information about the scope and meaning of the claim.” *Electro Med. Sys., S.A. v. Cooper Life Scis., Inc.*, 34 F.3d 1048, 1054 (Fed. Cir. 1994) (citations omitted). “[T]he specification is always highly relevant to the claim construction analysis. Usually, it is dispositive; it is the single best guide to the meaning of a disputed term.” *Vitronics*, 90 F.3d at 1582. “However, claims are not to be interpreted by adding limitations appearing only

1 in the specification.” *Id.* “Thus, although the specifications may well indicate that certain
2 embodiments are preferred, particular embodiments appearing in a specification will not be read
3 into the claims when the claim language is broader than such embodiments.” *Id.* Conversely,
4 “where [] the claim language is unambiguous, [the Federal Circuit has] construed the claims to
5 exclude all disclosed embodiments.” *Lucent Techs., Inc. v. Gateway, Inc.*, 525 F.3d 1200, 1215-
6 16 (Fed. Cir. 2008). “[T]he description may act as a sort of dictionary, which explains the
7 invention and may define terms used in the claims,” and the “patentee is free to be his own
8 lexicographer,” but “any special definition given to a word must be clearly defined in the
9 specification.” *Markman*, 517 U.S. at 989-90.

10 On the other hand, it is a fundamental rule that “claims must be construed so as to be
11 consistent with the specification.” *Phillips*, 415 F.3d at 1316. “The construction that stays true to
12 the claim language and most naturally aligns with the patent’s description of the invention will be,
13 in the end, the correct construction.” *Renishaw PLC v. Marposs Societa’ per Azioni*, 158 F.3d
14 1243, 1250 (Fed. Cir. 1998).

15 Finally, the court may consider the prosecution history of the patent, if in evidence.
16 *Markman*, 52 F.3d at 980. The prosecution history may “inform the meaning of the claim
17 language by demonstrating how the inventor understood the invention and whether the inventor
18 limited the invention in the course of prosecution, making the claim scope narrower than it would
19 otherwise be.” *Phillips*, 415 F.3d at 1317 (citing *Vitronics*, 90 F.3d at 1582-83); *see also Chimie*
20 *v. PPG Indus., Inc.*, 402 F.3d 1371, 1384 (Fed. Cir. 2005) (“The purpose of consulting the
21 prosecution history in construing a claim is to exclude any interpretation that was disclaimed
22 during prosecution.”) (internal quotations omitted).

23 In most situations, analysis of this intrinsic evidence alone will resolve claim construction
24 disputes. *Vitronics*, 90 F.3d at 1583. However, “it is entirely appropriate . . . for a court to consult
25 trustworthy extrinsic evidence to ensure that the claim construction it is tending to from the patent
26 file is not inconsistent with clearly expressed, plainly apposite, and widely held understandings in
27 the pertinent technical field.” *Pitney Bowes, Inc. v. Hewlett-Packard Co.*, 182 F.3d 1298, 1309
28 (Fed. Cir. 1999). Extrinsic evidence “consists of all evidence external to the patent and

prosecution history, including expert and inventor testimony, dictionaries, and learned treatises.” *Markman*, 52 F.3d at 980. All extrinsic evidence should be evaluated in light of the intrinsic evidence, *Phillips*, 415 F.3d at 1319, and courts should not rely on extrinsic evidence in claim construction to contradict the meaning of claims discernible from examination of the claims, the written description, and the prosecution history, *Pitney Bowes*, 182 F.3d at 1308 (citing *Vitronics*, 90 F.3d at 1583). While extrinsic evidence may guide the meaning of a claim term, such evidence is less reliable than intrinsic evidence. *Phillips*, 415 F.3d at 1318-19.

DISCUSSION

As an initial matter, the parties agree to following constructions, with respect to terms used in the ‘197 patent.

Object as used in Claims 1, 3, 6, 7 and 8 of the ‘197 Patent is defined as: **instance of a class**.

Object schema as used Claims 1, 3, 5, 6, 7 and 8 of the ‘197 Patent is defined as: **object-oriented view of one or more data store schemas**.

I. DISPUTED TERMS THAT DO NOT INVOLVE § 112(F)

A. Packing

An example of how “packing” is used in the ‘197 patent is demonstrated in Claim 1:

said first interface extracting the object attributes and the object name from the object to effect **packing** of the object attributes and the object name as data, said first interface **unpacking** the data to effect instantiating the object attributed and the object name into a new object,

Patent/Term	Oracle’s Proposal	Thought’s Proposal
Packing: Claims 1, 3, 7, 8 of ‘197 Patent.	translating data into a format suitable for communication and transport across a network	placing data into a form suitable for communication between software components

The parties agree the purpose of packing data is to make data suitable for communication. Oracle’s definition, however, seeks to incorporate two additional concepts: *translating* and *network*. There is no support for incorporating *translating* into the definition of packing. As an

1 initial matter, translation is a term that arguably requires its own definition. While Oracle argues
 2 that “packing” must do something to the data and that simply “placing” the data as suggested by
 3 Thought is too ambiguous, Oracle’s proposal to use “translating” itself imports further
 4 ambiguities. *See* Jagadish Resp. Decl., ¶ 25 (“Dr. Hosking provides a very cursory analysis of the
 5 term ‘unpacking’ in his declaration, and does not explain into what format or formats the data is
 6 translated.”). Moreover, data does not necessarily need to be “translated” for communication. It
 7 could, instead, be compacted, reordered, or otherwise manipulated for various purposes. As Dr.
 8 Jagadish notes, there are many purposes persons ordinarily skilled in the art would recognize as
 9 being served by packing data, including for storage in place, transport, orderliness or compaction.
 10 *See, e.g.*, Jagadish Resp. Decl., ¶¶ 16, 18. Oracle does not demonstrate that “translation” of data is
 11 necessary to achieve the agreed-to purpose here – communication – much less any of the other
 12 functions packing serves that are also identified in the specification, such as compression and data
 13 de-duplication.¹ *See* ‘197 Patent at 10:5-24; *see also* Hosking Decl. [Docket 71-1], ¶ 113
 14 (“packing process typically removes much of the metadata . . . and isolates the essential
 15 information”); Jagadish Deposition at 158:12-17 (packing puts “things into a simplified
 16 format”).

17 With respect to *network*, Oracle relies on examples in the specification of data being
 18 packed prior to being communicated across a network. *See* ‘197 Patent at 7:39-54 (“The
 19 communication medium 630 between the first adapter 400 and the second adapter 500 may
 20 comprise an Internet connection”); at 10:5-12 (“the subject invention reduces network latency
 21 (which is essential to reduce over the Internet and other io-bound and high latency networks) and
 22 increases the performance and scalability of the involved clients and servers”). Oracle argues
 23 these descriptions of the invention make it clear that the “purpose” of the invention is to improve
 24 the performance of distributed client-systems. *See* Hosking Decl., ¶ 112.

25
 26 ¹ In support of its argument that a person of ordinary skill in the art would understand the function
 27 of “packing” data to be communication of data across a network, Oracle’s expert, Dr. Hosking,
 28 relies on various prior art publications which disclose packing of data prior to its transmission
 across networks. Declaration of Anthony Hosking, Ph.D, in Support of Oracle’s Response
 (Docket No. 71-1), ¶¶ 116-125. However, nothing in those prior art references or Dr. Hosking’s
 declaration establish that “packing data” *necessarily* implicates transporting data across networks.

1 However, there are other examples described in the specification showing implementation
 2 of the invention in a non-networked, two tier environment. Specifically, the ‘197 Patent 4:16-21
 3 (emphasis added) explains:

4 the subject invention is capable of accessing objects distributed over
 5 a three tier environment using exactly the same application program
 interface (“API”) 700 as the **local two-tier environment**.

6 *See also* ‘197 Patent 4:49-52 (“One embodiment of the subject invention is implemented using the
 7 adapter technology and is therefore capable of being deployed as either a 2-tier or as a 3-tier
 8 environment transparently.”).²

9 Oracle is correct that one claimed benefit from a networked implementation of the
 10 invention – as described in the specification – is network efficiency (‘197 Patent at 10:5-12;
 11 10:17-24). But Oracle has not shown that the claim language or other parts of the specification
 12 *require* implementation in a networked system. Reducing network latency is a significant benefit
 13 of the invention, depending on the embodiment being used, but mentioning that significant benefit
 14 did limit the scope of the invention *itself*. *But see Verizon Servs. Corp. v. Vonage Holdings Corp.*,
 15 503 F.3d 1295, 1308 (Fed. Cir. 2007) (“When a patent thus describes the *features* of the ‘present
 16 invention’ as a whole, this description limits the scope of the invention.” (emphasis added)).
 17 Moreover, as described in the Summary, the purpose of the invention was not simply reducing
 18 network latency, but also solving the object-relations impedance mismatch; a significantly
 19 different problem. ‘197 Patent 3:29-36.

20 Finally, while Oracle argues that the “logical division” (Fig. 1 at 650) separating the first
 21 interface/adaptor from the second interface/adaptor must be a network division, Oracle points to
 22 nothing in the claim language or specification that defines a “logical” division. The term “logical”
 23 is significantly distinct from the “physical” division proposed by Oracle. Jagadish Resp. Decl. ¶

24
 25 ² At the claim construction hearing Oracle asserted that a 2-tier system – references to which in
 26 the specification Thought relies on in support of its construction – can itself be networked but a
 27 “local” system (a term also used in the specification) is a much more limited system. However,
 28 Dr. Hosking did not discuss this issue in his declarations, and Oracle cites to no evidentiary
 support for this argument. Dr. Jagadish disagrees: “A person of ordinary skill in the art would
 understand that the phrase “the local two-tier environment” is an example in which the invention
 is implemented locally on just one computer, and not in a distributed network configuration, such
 as the distributed three-tier embodiment also shown above.” Jagadish Resp. Decl. ¶ 21.

15. To a person of ordinary skill in the art a logical division simply means that software on one side of a logical divide can be programmed independent of the actions of the software on the other side. *Id.* ¶¶ 15-16. There is nothing in the language of the claims, or in the specification, that *requires* packed data to be communicated across a network and I will not incorporate that limitation into the definition of packing.³

Therefore, **packing** is defined as: **placing data into a form suitable for communication between software components.**

B. Unpacking

Patent/Term	Oracle's Proposal	Thought's Proposal
Unpacking: Claims 1, 3, 7, 8 of the '197 Patent	translate/translating packed data into another format	retrieving data from a packed data structure

Both proposals flow from the parties' definitions of "packing." While Oracle argues that Thought's "packed data structure" incorporates a new term that itself must be defined, I find that packed data structure is readily understandable and logically flows from the adopted definition of packing.

As such, **unpacking** is defined as: **retrieving data from a packed data structure.**

C. Interface

For example, as used in Claim 1:

said first **interface** extracting the object attributes and the object name from the object to effect packing of the object attributes and the object name as data, said first **interface** unpacking the data to effect instantiating the object attributed and the object name into a new object, and a second **interface** in communication with said first **interface** and in communication with at least one data store, said second **interface** have a meta data map comprising . . .

³ As noted above, the specification contemplates an embodiment of the invention operating in a local, two-tier system. The requirement of networked or distributed systems under Oracle's proposed definition would exclude the local embodiment, something which is "rarely, if ever, correct and would require highly persuasive evidentiary support. . . ." *Vitronics Corp. v. Conceptronic*, 90 F.3d 1576, 1583 (Fed. Cir. 1996).

Patent/Term	Oracle's Proposal	Thought's Proposal
Interface: Claims 1, 2, 7 of '197 Patent	software component that facilitates communication between at least two distributed software systems	part of an adapter abstraction layer that facilitates interaction between software components

Oracle is attempting to incorporate the limitations that the interface communicates between *distributed software systems*. Thought responds that there are no justifications to import those limitations into the definition of interface. As discussed above with respect to networks and the logical division, while there is some support for Oracle from the discussion of preferred embodiments in the specification that the invention is capable of communicating across distributed systems, there is no reason for me to limit the claims *to* distributed systems. The specification itself discloses that “the subject invention is capable of accessing objects distributed over a three tier environment using exactly the same application program interface (“API”) 700 as the local two-tier environment. This is accomplished by using access “adaptive technology comprising the adapter abstraction layer 600.” ‘197 Patent 4:16-22. The interface contemplated is used in the two tier (local) and three tier (distributed) systems.

Oracle also argues that the common understanding of interface is that an interface is usually used to describe a template for interactions between software components and not a component itself. Hosking Decl. ¶¶ 131-132, 134. However, in support of its position that interface should be defined as a “software component,” Oracle argues that Thought specifically defined interface to be the same as an “adapter,” which both parties agree is a software component. Oracle relies on language from claim 2 of the ‘197 Patent where Thought claims “A system according to claim 1 wherein the first interface comprises a first adapter and the second interface comprises a second adapter.” ‘197 Patent 35:23-25.

Neither Thought nor its expert addresses this argument, except to point to other language in claims 1 and 7 explaining that the interfaces are part of the adapter abstraction layer. ‘197 Patent claim 1 35:7-17 (“an adapter abstraction layer having a first interface . . . and a second interface in communication”); claim 7 36:66-67 (“an adapter abstraction layer having a first

interface and a second interface”). At the claim construction hearing, Thought asserted that claim 2 simply makes it clear where the “adapters” are located (in the adapter abstraction layer, with the first and second interfaces for the first and second adapters respectively) and that, for example, claim 3 clarifies the purpose of the abstraction adapter layer and its use of adapters. *See also* Jagadish Resp. Decl. ¶ 47 (“In the case of the ‘197 patent, the first adapter implements the first interface, and the second adapter implements the second interface.”).

I find that while claim 2 could be read as using the terms interface and adapters interchangeably (“wherein said first interface comprises a first adapter and the second interface comprises a second adapter”), it can also be read as identifying where the adapters are located (as argued by Thought) and as adding more functionality to the invention. As the Court is required to give effect to a patent’s use of different terms, unless contrary to the express claim language, Thought’s proposed construction is more appropriate.

As Thought’s expert explains, interfaces are commonly understood by persons ordinarily skilled in the art as “part of an adapter abstraction layer that facilitates interaction between software components.” Jagadish Resp. Decl. ¶ 27. This is supported by the specification which explains a distinction between the adapters and interfaces: in the present invention, because the adapters in the adapter abstraction layer “implement the same API [interface] 700, they are interchangeable in the object application 101, providing new and useful functionality. . . .” ‘197 Patent at 9:38-42.

Therefore, **interface** is defined as: **part of an adapter abstraction layer that facilitates interaction between software components.**

D. Adapter

For example, as used in Claim 3:

a first **adapter** responsive to the object application including an application bridge . . . said first **adapter** extracting the object attributes and the object name . . . said first **adapter** unpacking the data to effect instantiating the object attributes and the object name into a new object; and a second **adapter** in communication with said first adapter and in communication with at least one data store, said second **adapter** having a meta data map

Patent/Term	Oracle's Proposal	Thought's Proposal
Adapter: Claims 3, 8 of '197 Patent	interchangeable software component that facilitates communication between at least two distributed software systems	runtime interchangeable part of an adapter abstraction layer that enables incompatible software components to work together

The parties agree that the adapters are interchangeable, part of the abstraction layer, and facilitate communication. Oracle, again, seeks to incorporate the limitation that the components are communicating between two distributed software systems. For same reasons as discussed above, I find no support for Oracle's proposed "two distributed systems" limitation. *See also* Hosking Deposition, Vol II, 150:4-22 (acknowledging that invention of '197 Patent will function even if there is no physical separation, but just a logical separation between the first and second adapters).

Thought's definition includes that the adapters are interchangeable at "runtime." There is significant support for Thought's position. The specification repeatedly refers to the invention as incorporating "runtime" adapters. *See, e.g.*, '197 Patent at 2:46-49 ("Accordingly, there is a need to dynamically build the code for accessing the data store at runtime. . . ."); 4: 66-67 ("The subject invention uses the adapter technology to dynamically load data store access code at runtime."); 9:34-47 ("The present invention comprises the adapter abstraction layer 600 comprising a set of runtime adapters (e.g. the first adapter 400, the second adapter 500, the n-th adapter 4XX, 5XX), which can be transparently interchanged by the object application 101. . . . Because all adapters implement the same API 700, they are interchangeable in the object application 101, providing new and useful functionality, even beyond the subject implementation, without requiring object application 101 modification. Object application 101 runtime flags can instantiate entirely new adapters which might interface with radically different data store(s) 302 without modification to object application 101."); 10:25-33 ("New adapters . . . may be added to the adapter abstraction layer 600 dynamically, even at runtime, as long as they comply with the API.")

As Oracle points out, "runtime" is not used in the claim language. Dr. Jagadish explains

that runtime adapters are “inherent in the adapters of the invention claimed in the ‘197 patent.” Jagadish Resp. Decl. ¶ 41. He describes the importance of runtime changeable adapters and runtime flags that signal their use, in that that “runtime flags as mentioned in the ‘197 patent specification allow different runtime interchangeable adapters to be used without having to recompile the application. In other words, there is no need to reenter the compile time phase in order to effect interchanging the runtime interchangeable adapters of the ‘197 patent.” *Id.* ¶ 44. He says that “[s]ignificant time and costs are often associated with modifying an application’s source code and subsequently recompiling the application. The ability to use or substitute various runtime interchangeable adapters without having to recompile the application provides greater flexibility with respect to data sources while reducing the time and cost required to maintain the application.” *Id.* at 45.⁴ This comports with the explanation of the invention in the background section “there is a need to dynamically build the code for accessing the data store at runtime, based on the object and its attributes, as opposed to embedding the access code within the object.” ‘197 Patent 2:46-49; *see also Renishaw PLC v. Marposs Societa’ per Azioni*, 158 F.3d 1243, 1250 (Fed. Cir. 1998) (“This explanation also aligns with the summary of the invention as described. The construction that stays true to the claim language and most naturally aligns with the patent’s description of the invention will be, in the end, the correct construction.”). Finally, Dr. Jagadish opines that Oracle’s proposal – defining adapters as simply components that facilitate communication – does not sufficiently distinguish adapters from interfaces. *Id.* ¶ 46.

Therefore, **adapter** is defined as: **runtime interchangeable part of an adapter abstraction layer that enables incompatible software components to work together.**

E. Adapter abstraction layer

For example, as used in Claim 1:

⁴ Dr. Hosking does not dispute Dr. Jagadish’s opinion or address it in his reply declaration. Instead, he focuses on the mention of runtime flags in the specification as used to instantiate and deploy new adapters, arguing that runtime flags are parameters or indicators “that a program evaluates once and only once each time the program executes—at the moment the program is invoked.” Hosking Response Decl. ¶ 150. However, that discussion does not undermine or address the key issue; whether the adapters are necessarily interchangeable at runtime as described in the ‘197 Patent.

an **adapter abstraction layer** having a first interface responsive to the objection application including an application bridge . . .

As used in Claim 7:

communicating a request . . . from the object application to an **adapter abstraction layer** having a first interface and a second interface . . . communicating the request and at least one such new object and/or status from the **adapter abstraction layer** to the application program.

Patent/Term	Oracle's Proposal	Thought's Proposal
Adapter abstraction layer: Claims 1, 7 of '197 Patent	software components with interfaces provided by adapters	software for accessing at least one data store from at least one object application, having interfaces provided by runtime interchangeable adapters

Both sides agree that the interfaces are provided by the adapters and, as discussed above, that adapters are interchangeable. At the claim construction hearing, Thought agreed that Oracle's proposed definition would be appropriate if the Court adopted Thought's proposed definition of adapter.

Therefore, **adapter abstraction layer** is defined as: **software components with interfaces provided by adapters.**

F. Fully synchronized caching system

As used:

A local or distributed computer system comprising fully synchronized caching system utilizing at least one computer implemented program that synchronizes the caching of a delegated data source access management object to relational mapping layer and the transaction coordination facilities of an application server

Patent/Term	Oracle's Proposal	Thought's Proposal
Fully synchronized caching system: Claim 7 of '956 Patent	caching system in which any query for a value obtains only the most recently committed value	caching system in which any changes to the cache will automatically be synchronized with the primary data source or record upon transaction

		completion
--	--	------------

Oracle and its expert spend much time explaining why its proposed construction is consistent with the understanding a person of ordinary skill in art had about “weakly” and “strongly” consistent synchronization, and how “fully” is most likely synonymous with “strongly.” Hosking Decl. ¶ 168. However, the claim language itself defines the scope and purpose of the “fully synchronized caching system,” namely that the system:

provides the feature that any changes to the cache will automatically be synchronized with the primary data source or record upon transaction completion, including commit or roll-back of changes for both data sources.

‘956 Patent at 51:6-10, 25-29.

This intrinsic explanation of the caching system is more significant than Oracle’s extrinsic discussion of what a person ordinarily skilled in the art understands about weakly or strongly consistent caching.⁵ Dr. Hosking argues that adopting this definition of caching as “fully synchronized” is incorrect because it does not give any meaning to the word “fully” and it “merely describes a particular aspect of synchronization, and even then it does so incompletely” because it does not explain what happens when and if the primary source is modified. Hosking Decl. ¶¶ 171-172. Dr. Jagadish agrees that “fully” correlates to a “strongly” consistent caching system, but that the strongly consistent nature of the caching is adequately captured in Thought’s proposal which explains that the cache will “automatically” be synchronized for “any” changes. Jagadish Resp. Decl. ¶ 56. However, Dr. Jagadish persuasively argues that Oracle’s proposed construction “fails to capture two important aspects of the caching system, namely, (1) the caching system includes a primary data source and at least one cache data source, and (2) the cached data in the cache data source have their values synchronized against the corresponding data in the primary data source.” *Id.* ¶ 57. As Dr. Jagadish points out, Thought’s definition – taken from the claim language itself – explains when the caching occurs (“upon transaction completion”) and that the cache will automatically be synchronized with the primary data source. *Id.* ¶ 58.

⁵ As Thought’s expert points out, there are many other types of caching systems in addition to weakly and strongly consistent, and therefore Dr. Hosking’s analysis is oversimplified. Jagadish Resp. Decl. ¶ 54.

Therefore, **fully synchronized caching system** is defined as: **a caching system in which any changes to the cache will automatically be synchronized with the primary data source or record upon transaction completion.**

II. TERMS WHICH ORACLE ALLEGES ARE INDEFINITE UNDER § 112(B)

Oracle argues that five claim terms are impermissibly indefinite and, therefore, invalid. The parties dispute whether Oracle's indefiniteness challenge is appropriate resolved on claim construction (as opposed to on a motion for summary judgment) and what the burden is on Oracle to demonstrate indefiniteness at this juncture.

A. Legal Standard

Indefiniteness is a part of 35 U.S.C. § 112, therefore, the mandates of 35 U.S.C. § 282 apply. *See* 35 U.S.C. § 282(b)(2). Under 35 U.S.C. § 282(a), the burden of establishing invalidity for indefiniteness falls squarely on Oracle, the party asserting invalidity. *Microsoft Corp. v. i4i Ltd. P'ship*, 131 S. Ct. 2238, 2242, 2011 U.S. LEXIS 4376, * 7 (2011) (invalidity defenses must be proved by "clear and convincing evidence"); *see also Nautilus, Inc. v. Biosig Instruments, Inc.*, 134 S. Ct. 2120, 2014 U.S. LEXIS 3818, * 24 fn.10 (June 2, 2014).⁶ Oracle's showing, however, cannot be defeated by Thought raising a genuine issue of material fact as to indefiniteness (and thus validity). Instead, because indefiniteness is a question of law for the court to determine, I must resolve any disputed issue of material fact based on the evidence before me.⁷

⁶ Oracle contends that it need not prove indefiniteness, which is purely a question of law, by clear and convincing evidence. *See, e.g.,* Sur-Reply at 1, n.1 (and cases cited therein). However, the Supreme Court's citation to *Microsoft Corp. v. i4i's* holding that "invalidity" defenses must be shown by clear and convincing evidence in its recent decision addressing indefiniteness supports this Court's conclusion that even though indefiniteness is a question of law, the clear and convincing standard still applies to any facts Oracle contends support that determination. *See Nautilus, Inc.*, 2014 U.S. LEXIS 3818 at * 24 fn.10.

⁷ Thought argues that a finding that claims are invalid for indefiniteness would effectively grant summary judgment in favor of Oracle. Therefore, Thought contends that its expert can raise material issues of fact precluding summary judgment. Thought's Claim Construction Brief at 30-31. However, the cases Thought relies on to argue that material issues of fact can preclude summary judgment on indefiniteness do not support its contention. In *Rembrandt Data Techs., LP v. AOL, LLC*, 641 F.3d 1331, 1342 (Fed. Cir. 2011), the Federal Circuit simply noted that the district court failed to address whether plaintiff's expert raised a genuine issue of fact about the knowledge of skilled artisans, not whether the district court was precluded from resolving a dispute of fact. Similarly in *Aristocrat Techs. Austl. Pty Ltd. v. Multimedia Games, Inc.*, 266 Fed. Appx. 942, 945, 2008 U.S. App. LEXIS 3774, 7 (Fed. Cir. 2008), the case was remanded so that the district court could *resolve* the material issues of fact regarding whether one skilled in the art

As the Supreme Court recently explained “a patent is invalid for indefiniteness if its claims, read in light of the specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 2014 U.S. LEXIS 3818, * 6. To evaluate an indefiniteness claim I must: (i) evaluate the claim from the perspective of someone skilled in the relevant art; (ii) read the claim in light of the patent’s specification and prosecution history; and (iii) measure definiteness from the viewpoint of a person skilled in the art at the time the patent was filed. *Id.*, 2014 U.S. LEXIS 3818 at *17-18. The claim language must be “precise enough to afford clear notice of what is claimed, thereby “‘appris[ing] the public of what is still open to them.’” *Id.* (quoting *Markman v. Westview Instruments*, 517 U.S. 370, 373 (1996) (internal quotation omitted); see also *Carnegie Steel Co. v. Cambria Iron Co.*, 185 U. S. 403, 437 (1902) (“any description which is sufficient to apprise [steel manufacturers] in the language of the art of the definite feature of the invention, and to serve as a warning to others of what the patent claims as a monopoly, is sufficiently definite to sustain the patent”).

B. Terms At Issue

1. Extensive Knowledge

For example, as used in Claim1:

the mapping system portion provides an interface . . without requiring the user to have **extensive knowledge** of a particular relational database as a source of the data, or **extensive knowledge** about how to directly access that relational database.

Patent/Term	Oracle’s Argument	Thought’s Construction
Extensive knowledge: Claims 1, 3, 11 of ‘956 Patent	Indefinite under 112(b)	knowledge of the schema and query language sufficient to save or retrieve data

Oracle argues “extensive knowledge” is indefinite because it is not and cannot be

would identify structure in the specification. Instead, because claim indefiniteness is a question of law, and one on which the court may refer to expert extrinsic evidence, the court can and must resolve factual disputes. See *Exxon Research & Eng’g Co. v. United States*, 265 F.3d 1371, 1376 (Fed. Cir. 2001).

accurately defined by a person ordinarily skilled in the art. Thought contends that “extensive knowledge” is not indefinite because “because the environment dictates the necessary preciseness of the terms.” *Power-One, Inc. v. Artesyn Techs., Inc.*, 599 F.3d 1343, 1348 (Fed. Cir. 2010) (rejecting claim that “near” and “adapted to” are indefinite because of clarity provided by remaining claim language). Here, Thought contends that “extensive knowledge” is clarified by the language following the term: “extensive knowledge of a particular relational database as a source of the data, or extensive knowledge about how to directly access that relational database.” ‘956 Patent 49:52-54. It is also clarified by the description of an embodiment which explains its purpose is to manage database access code “requiring only an understanding of the data store 302 and the data store schema 300, instead of requiring object programming knowledge within the modern working environment.” ‘956 Patent 11:30-35; *see also id.*, 10:32-35 (allows engineers to “concentrate on applications code without concern for or need to learn SQL or any other native access code 205 of the data store.”); *id.* 26:50-54 (“translation system . . . provides an entire set of tools to simplify and improve an operators ability to manipulate maps without requiring a high level of programming knowledge or database query language familiarity. . . .”).

Dr. Jagadish explains that in order to use a database, after gaining access through a password or other code, a user would have to know the schema and query language of the database in order to edit and retrieve data. Jagadish Resp. Decl. ¶ 66. He declares that “person of ordinary skill in the art would understand the term ‘extensive knowledge’” of a particular relational database or how to directly access that relational database, to mean “‘knowledge of the schema and query language sufficient to save or retrieve data’ in the context of the ‘956 patent.” *Id.* ¶ 61. Oracle’s expert, Dr. Hosking disputes that assertion, arguing instead that knowledge of “access” could also mean the technicalities of figuring out a database location on the network or any credentials necessary to access the database. Hosking Decl. ¶ 189. However, when viewed in the context of the claim language and specification, Dr. Jagadish’s position is more persuasive.

I find that “extensive knowledge” is not indefinite when considered with the remaining claim language and the descriptions of the invention/embodiments in the specification. When read in light of these, a person ordinarily skilled in the art would understand that extensive knowledge

means that the user can perform certain tasks – as explained in the patent, including editing or creating tables, fields, or maps – without requiring the user to have extensive knowledge of a relational database or how to directly access that database. Jagadish Decl. ¶ 64.

The case on which Oracle relies is inapposite. In *Datamize, LLC v. Plumtree Software, Inc.*, 417 F.3d 1342, 1345 (Fed. Cir. 2005), the court concluded that claim language regarding displays of electronic information on kiosks “in conformity with a desired uniform and aesthetically pleasing look and feel for said interface screens on all kiosks of said kiosk system,” was indefinite. The court found that “Datamize has offered no objective definition identifying a standard for determining when an interface screen is ‘aesthetically pleasing.’ In the absence of a workable objective standard, ‘aesthetically pleasing’ does not just include a subjective element, it is completely dependent on a person’s subjective opinion.” *Id.* at 1350. Because “[t]he scope of claim language cannot depend solely on the unrestrained, subjective opinion of a particular individual purportedly practicing the invention,” the court found the term “aesthetically pleasing” indefinite. *Id.* Here, however, the specification and description of the invention give persons skilled in the art an objective standard to determine the knowledge required to practice the invention. Unlike in *Datamize*, the scope of the language does not depend on the subjective opinion of the person practicing the invention and the patent is not indefinite.⁸

Moreover, while “extensive” knowledge is a word of degree, words of degree are not indefinite where the “environment,” meaning the remaining claim language and specification, “dictates the necessary preciseness of the terms.” *Power-One, Inc.*, 599 F.3d at 1348; *see also Interval Licensing LLC v. AOL, Inc.*, 2014 U.S. App. LEXIS 17459 (Fed. Cir. Sept. 10, 2014) (“Claim language employing terms of degree has long been found definite where it provided

⁸ This case is also unlike *Interval Licensing LLC v. AOL, Inc.*, 2014 U.S. App. LEXIS 17459 (Fed. Cir. Sept. 10, 2014), where the Federal Circuit held that the phrase “unobtrusive manner” was highly subjective and provided little guidance to one skilled in the art because the claim language lacked “objective boundaries.” *Id.* at *15-16. Here, the context of the claim language and the specification provide one skilled in the art with clarity on what is covered. *But see id.* at *19 (“we find that the specification is at best muddled, leaving one unsure of whether the ‘unobtrusive manner’ phrase has temporal dimensions as well as spatial dimensions. The hazy relationship between the claims and the written description fails to provide the clarity that the subjective claim language needs.”).

enough certainty to one of skill in the art when read in the context of the invention.”); *Seattle Box Co. v. Indus. Crating & Packing, Inc.*, 731 F.2d 818, 826 (Fed. Cir. 1984) (“when a word of degree is used the district court must determine whether the patent’s specification provides some standard for measuring that degree.”). That is exactly what the specification does in this case; it indicates that a user need not have extensive knowledge of the structure and contents of a relational database that would otherwise be necessary to access and change the information in that relational database.

However, to add additional clarity for purposes of claim construction, **extensive knowledge** is further defined as: **knowledge of the schema and query language sufficient to save or retrieve data.**

2. To provide . . . to a system user who is

As used in Claim 1 of the ‘956 Patent:

the mapping system portion is designed **to provide to a system user who is** accessing, creating or updating maps, or accessing objects on a system to make data changes related to a particular object and to . .

Patent/Term	Oracle’s Argument	Thought’s Construction
To provide . . . to a system user who is: Claim 11 of ‘730 Patent; Claim 1 of ‘956 Patent	Indefinite under 112(b)	to provide a system user the ability to make data changes related to a particular object as either local or global changes on the computer system

Oracle argues that the “thing” being provided to a system user is missing and, therefore, indefinite. Oracle claims that Thought’s definition is an impermissible attempt to correct an incomplete phrase or drafting error in patent. As the Federal Circuit has explained, a district court can act to correct an error “in a patent by interpretation of the patent where no certificate of correction has been issued,” but can only do so “if (1) the correction is not subject to reasonable debate based on consideration of the claim language and the specification and (2) the prosecution history does not suggest a different interpretation of the claims.” *Novo Indus., L.P. v. Micro*

1 *Molds Corp.*, 350 F.3d 1348, 1354 (Fed. Cir. 2003); *see also Hoffer v. Microsoft Corp.*, 405 F.3d
2 1326, 1331 (Fed. Cir. 2005) (where PTO erred in failing to change reference to a specific
3 antecedent claim number, court held that where “the error was apparent from the face of the
4 patent, and that view is not contradicted by the prosecution history.”).⁹ Oracle asserts that the
5 correction Thought is suggesting is subject to debate, because in addition to “an ability” as
6 suggested by Thought, “the claims of these patents call for the providing of interfaces, software
7 logic, features, and programming applications, among other things. Each of these terms, as well
8 as others, would make sense in the context of ’956 claim 1 or ’730 claim 11.” Hosking Decl. ¶
9 192.

10 Thought responds that it is not seeking to correct an error, but to interpret the claim
11 language consistently with the remaining claims and the specification by clarifying that what is
12 provided to the system user is the ability to do the items listed in the claim language. ‘956 Patent
13 49:43-45 (“to make data changes related to a particular object and to promulgate the changes to
14 that object”); ‘730 Patent 53:16-18 (“to make data changes related to a particular object and to
15 promulgate the changes to that object”); *see also* Jagadish Resp. Decl. ¶¶ 77-78 (“if the claim
16 limitation is considered based on the specifications of the ‘956 and ‘730 patents, then it is clear
17 that the claim limitation already recites what is being provided to the system user, namely the
18 ability or capability ‘to make data changes related to a particular object and to promulgate the
19 changes to that object as either local or global changes on the computer system.’ . . . It is only
20 logical that the claim limitation would recite this capability of the mapping system as what is
21 being provided to the system user, and a person of ordinary skill in the art would have no
22 difficulty understanding the meaning of the claim limitation.”).

23 When the claim language is read in full, I agree with Thought that the claim is conferring
24 “the ability” of a user to do the things listed in the claim; make data changes related to a particular
25 object, etc. This is not, as Oracle argues, a correction of an “error.” It is instead a construction
26 clarifying the awkward language of the claim and a construction that flows directly from the text
27

28 ⁹ Any other change must be sought from the Patent Office. *See* 35 U.S.C. §§ 254-55.

of the claims themselves.

Therefore, **to provide . . . to a system user who** is defined as: **to provide a system user the ability to make data changes related to a particular object as either local or global changes on the computer system.**

3. Object

Class is specifically defined in the '600 and '862 Patents as:

"Object" as used in the context of this document is a general term referring to a logic unit in a computer application or a computer software program. . . . The term "object" may be used interchangeably with the term "class" as a template or as an instance depending on the context.

Class is specifically defined in the '730 Patent as:

An organized set of encapsulated programming code designed to act on itself at the request of some external system, which system may pass in some additional information to the object in that request

Patent/Term	Oracle's Argument	Thought's Construction
Object: Claims 1, 6, 9, 13, 14, 17 of '600 Patent; Claims 1, 10, 14 of '862 Patent	Adopt express definitions of terms in patents; but those definitions are indefinite under 112(b)	Instance of a class
Object: Claims 1-7, 10-14, 16 of '730 Patent	An organized set of encapsulated programming code designed to act on itself at the request of some external system, which system may pass in some additional information to the object in that request	Instance of a class

a. '600 and '862 Patents

Oracle argues the express definition of object provided in '600 and '862 Patents should be adopted. The Patents define "object" as follows: "'Object' as used in the context of this document is a general term referring to a logic unit in a computer application or a computer

1 software program. . . . The term ‘object’ may be used interchangeably with the term ‘class’ as a
 2 template or as an instance depending on the context.” ‘600 Patent 4:1-7; *see also* ‘862 Patent
 3 3:63-4:2. However, Oracle argues that this definition is itself ambiguous – because it conflates the
 4 normally distinct concepts of “object” and “class” – and therefore, the term is indefinite in each
 5 case. *See, e.g.*, Hosking Decl. ¶ 197.

6 Thought’s proposed construction is narrower than the broad definition adopted in the
 7 patent specifications, because the specification-definition allows “object” to be used as both a
 8 “class” and instance of a class depending on the context of use. Because Thought has acted as its
 9 own lexicographer, I adopt Thought’s own definition. *Phillips v. AWH Corp.*, 415 F.3d 1303,
 10 1316 (Fed. Cir. 2005) (“the specification may reveal a special definition given to a claim term by
 11 the patentee that differs from the meaning it would otherwise possess. In such cases, the inventor’s
 12 lexicography governs.”). Therefore, for the ‘600 and ‘862 Patents, **object** is defined as: **a general**
 13 **term referring to a logic unit in a computer application or a computer software program,**
 14 **that may be used interchangeably with the term “class” as a template or as an instance**
 15 **depending on the context.**

16 I disagree with Oracle’s argument that the definition is inherently ambiguous. Both the
 17 definitions in the ‘600 and ‘862 Patents expressly rely on the *context* of each use of object in the
 18 Patents to inform a person skilled in the art whether in that case object is being used as an instance
 19 or as a template. Oracle points to no examples from the claim language or specification where the
 20 context does not provide adequate precision for a skilled person to be able to understand how
 21 object is being used.¹⁰ For example, in Claim 1 of the ‘600 Patent, the term “object” is further
 22 modified by “data object” or “object graphs model.” The context of those uses of “object” tell a

23
 24 ¹⁰ Oracle’s expert notes that in “many object-oriented languages, classes are themselves objects.”
 25 Hosking Decl. ¶ 198. Dr. Hosking explains that, nevertheless, the terms are not synonymous, but
 26 the use depends upon the *context*. “[A] particular class, such as the Filing class, is itself an
 27 instance of another class, typically called Class. The class Class is also an instance of itself. This
 28 may sound confusing, but is not unusual.” *Id.* Context, therefore, is key to determining the
 appropriate use of object versus class. While Dr. Hosking asserts that in his opinion “it is asking
 too much” of a person or ordinary skill in the art to use context to determine whether “object”
 means a class or an object, *id.* ¶ 203, he points to no examples in the Patents showing why that is
 so.

person skilled in the art what they need to know about the particular object. In the absence of specific examples where Oracle claims the context is insufficient, Oracle's indefiniteness argument is rejected.

b. '730 Patent

Oracle argues the explicit definition used in the '730 Patent at 17:60-65 should be adopted, and does not challenge that definition as indefinite. Oracle Response Br. at 23. In Reply, Thought does not object to that definition. Reply at 16:13-14.

Therefore, for the '730 Patent, **object** is construed as: **an organized set of encapsulated programming code designed to act on itself at the request of some external system, which system may pass in some additional information to the object in that request.**

4. Model

"Model" is also expressly defined in the patents as follows:

"Application model" or simply "model" are essentially interchangeable terms employed herein as abstractions to logically convey a collective description or other representation for It is important to understand the context . . . for the ease of communication abstractions of the model, possible implementations of the model and instances of the model are all referred to generally as "application model" or "model." From the context of its use the term will be clear.

Patent/Term	Oracle's Argument	Thought's Construction
Model: Claims 1, 7, 9, 10 of the '600 Patent; Claims 1, 8, 10, 11 of the '862 Patent	The patents expressly define the term; but those definitions are indefinite under 112(b)	logical or abstract representation or description

Oracle argues that the patents disavow the normal use of the term, and instead adopt an idiosyncratic one that, like object, depends upon its context. *See, e.g.*, '600 Patent 5:17-26 ("Ordinarily computer engineers refer to the model as an abstraction rather than a specific possibility or instance of the model as applied. However, in this document for the ease of communication abstractions of the model, possible implementations of the model and instances of the model are all referred to generally as application model or model. From the context of its use the term will be clear.")). Oracle contends that the definition of model as used in the patents is

insolubly ambiguous and indefinite because the patents provide no guidance on what the context should be or how it is applied to each use of model in the terms (abstractions, implementations, and instances). However, Dr. Jagadish explains that the context of the use of model adequately informs a person skilled in the art about the particular meaning of model. *See* Jagadish Resp. Decl. ¶¶ 69, 72 (“The ‘600 and ‘862 patents describe various types of models by adding specific qualifiers to the word “model,” such as object model, object graph model, complex data object graph model, CDOG model, and navigation model. The qualifiers make it clear what specific type of model it is in each instance.”). Oracle’s expert does not dispute this or point to any actual uses of “model” in the claims or specifications that do not provide adequate context.

Thought’s proposed definition is based on, but is not a verbatim adoption of, the express definition provided in the Claims. Thought proposes that model be defined as a “logical or abstract representation or description,” where the Patents define “‘Application model’ or ‘model’ are essentially interchangeable terms employed herein as abstractions to logically convey a collective description or other representation for a set of complex data objects and a corresponding description or other representation of their relationships.” ‘600 Patent 5:16-22. Other than arguing that the Patents’ use of “model” is indefinite, Oracle does not dispute or challenge the proposed, narrower definition suggested by Thought.

I find that **model** should be defined as: **logical or abstract representation or description**. I reject Oracle’s indefiniteness argument because a person of ordinary skill in the art would know from the context how model is being used in each instance.

5. Changes to a CDO or its relationship of a CDOG model

This term is used twice, as exemplified in Claim 1:

b) a feature providing an interface . . . or providing an interface for an editable input or source, such as a file, that can be modified to implement **changes to a CDO or its relationships of a CDOG model**.

Patent/Term	Oracle’s Argument	Thought’s Construction
Changes to a CDO or its relationship of a CDOG model:	Indefinite under 112(b)	changes to (1) a CDO, or (2) relationships of the CDO, the CDO

Claims 1, 6 of '481 Patent		and its relationships being part of a CDOG model
----------------------------	--	--

Oracle argues first that Thought's proposed construction is illogical because, a "CDOG is 'an abstraction to logically represent a set of complex data objects and a set of their corresponding relationships'" ('481 Patent 5:5-8), and a model is "a representation of relationships among classes." When these two terms are used together, "the plausible understanding is that the higher level of abstraction applies, and that a 'CDOG model' is a representation of relationships among classes, the instances of which may be related in a CDOG." Hosking Decl., ¶ 211. As such, according to Dr. Hosking and contrary to Thought's proposal, "neither a CDO nor its relationships are part of a CDOG model, because a CDOG model represents classes and their relationships and not objects and their relationships." *Id.* ¶ 212.

Oracle also argues that something is missing from the claim language itself (presumably because of a mistake by the patent drafter) and that Thought's proposed definition is an impermissible attempt to make a correction to otherwise unintelligible language. *See, e.g., Novo Indus., L.P. v. Micro Molds Corp.*, 350 F.3d 1348, 1354 (Fed. Cir. 2003). Oracle contends that Thought's attempted correction is impermissible because there is more than one way the language could reasonably be corrected. Dr. Hosking explains, this term could be rewritten as either (i) "changes to a CDO, or its relationships, or a CDOG model" or (ii) "changes to a CDO or its relationships to a CDOG model." Hosking Decl. ¶¶ 214-15. Because there are different options as to how to redefine the language of the claim, and those different options have a different scope and different implications, Thought can only seek a clarification from the PTO to correct the error with the language.

As to the first point, Thought argues that Dr. Hosking ignores that "model" is explicitly defined as not only an "abstraction," but also as an implementation and instance of the model, depending on the context. '481 Patent, 5:29-33. As explained in the summary of the invention, "An object of the present invention is to provide a system for creating, maintaining, accessing, navigating and persisting complex data objects as a complex data object graph (CDOG) model."

‘481 Patent 5:53-56. Therefore, as used in this patent, model is not simply a representation of classes and their relationships as Dr. Hosking contends but can also encompass “a representation of objects and their relationships,” Jagadish Resp. Decl. ¶ 81.

As to the second, Thought argues that its proposal is not an attempt to correct an error but instead a construction that clarifies – based on the express definitions of CDO, CDOG and model in the specification – what is otherwise awkward language. *Id.* ¶ 82.

Looking to the claim language and specification, I agree with Thought that the phrase “changes to a CDO or its relationship of a CDOG model” is not indefinite. Instead, reviewing how the phrases CDO, CDOG and CDOG model are actually used in the specification, Thought’s clarifying construction is appropriate. Therefore, **changes to a CDO or its relationship of a CDOG model** is defined as **changes to (1) a CDO, or (2) relationships of the CDO, the CDO and its relationships being part of a CDOG model.**

6. The information of (a)

As used in Claim 12:

- a) reading a source programming object logic model or a database definition file in a format selected from the group consisting of a UML data file, a XMI data file, and a XML file; and
- b) converting the information of (a) into a target member selected from the group consisting of a database definition XML file, a database mapping definition file, and a CDOG definition file.

Patent/Term	Oracle’s Argument	Thought’s Construction
The information of (a): Claim 12 of the ‘481 Patent	Indefinite under 112(b)	information read from a source programming object logic model or a database definition file in a format selected from the group consisting of a UML data file, a XML data file, and a XML file ((a) refers to limitation 12-(a))

Oracle argues that this phrase is indefinite, relying on the fact that in a prior claim chart, Thought itself identified the “information of (a)” as the information listed in claim 7(a). Hosking

Decl. ¶ 220. Dr. Hosking says reliance on 7(a) to provide the “information” for 12(b) makes sense because claim 12 depends from claim 7 and claim 7(a) explicitly cites “information” (e.g., information sufficient to construct a CDOG model) whereas 12(a) does not expressly cite information (e.g., reading a source programming object logic model or database definition file in various formats). Hosking Decl. ¶¶ 218, 221. But he argues because 12(a) cites “implicit” information, and 7(a) also has 7(a)(II) and 7(a)(III) subparts that likewise contain “implicit information,” what the 12(b) claim is actually referring to “as the information of (a)” is insolubly ambiguous. *Id.* ¶¶ 217, 222.

Thought responds that its prior reliance on 7(a) as the source of 12(b)’s “information of (a)” was a typographical error in its prior claim chart. Thought Reply Br. at 19, n.72. When arguing this issue on the merits, Thought asserts it has been consistent that “the information of (a)” refers to the information in 12(a), which immediately precedes the language in 12(b), which is logical and supported by the specification.

Reviewing the claim language and specification, I agree with Thought. Not only does 12(a) immediately precede 12(b), but the description of the invention in the specification supports that construction. *See* ‘481 Patent 6:54-63 (“A further object of the present invention is to a software tool capable of reading a source programming object logic model or a database file in a format selected from the group consisting of a UML data file, m [sic] a XMI data file, and a XML file and converting the information into a target member selected from the group consisting of a database definition XML file, a database mapping definition file, and a CDOG definition file. In a preferred object, the software can automatically generate a persistence layer that corresponds to the object model information of the source file.”); *see also* ‘481 Patent 9:16-29. As Dr. Jagadish explains, this section of the specification tracks the language of 12(a) and (b) but does not refer to any of the explicit or implicit information recited in Claim 7 that Dr. Hosking relies on. Jagadish Resp. Decl. ¶ 86.

Therefore, **the information of (a)** is defined as: **information read from a source programming object logic model or a database definition file in a format selected from the group consisting of a UML data file, a XML data file, and a XML file ((a) refers to limitation**

12-(a)).

7. **Wherein repository maps are organized according to at least one navigation model or schema**

As used:

(b) the computer system comprises a mapping system that includes logic for accessing each schema corresponding to a different data source accessible to the mapping system, . . . **wherein repository maps are organized according to at least one navigational model or schema**, or the mapping system provides

Patent/Term	Oracle's Argument	Thought's Construction
Wherein repository maps are organized according to at least one navigation model or schema: Claim 1 of the '730 Patent	Indefinite under 112(b)	[construed according to construction of constituent terms and common and ordinary meanings]

Oracle argues that because “repository maps” and “navigational models” or “navigation schemas” play no defined role in furthering the purpose of the claim or the invention, and because “repository maps has no generally understood meaning in the art,” the terms are indefinite. Hosking Decl. ¶¶ 231, 233, 234. Oracle also argues that repository maps and a navigational model are distinct, but the claim language fails to explain how they work together. As Dr. Hosking states, even accepting the definitions of the discrete terms used in other parts of the claim language and specification, “it is still the case that the patent gives no guidance to one of ordinary skill in the art as to how to organize one thing (in this case, ‘repository maps’) according to a model of another thing (‘in this case object relationships as represented in a “navigational model or schema”’).” Hosking Decl. ¶ 237.

Thought responds that repository maps are maps in the “mapping repository” which is part of the “mapping system,” and the specification describes exactly how repository maps can be imported into or exported from the mapping repository. *See* Jagadish Resp. Decl. ¶ 88 (“That is, they are the rules governing how data are mapped between two or more data sources. Without

these maps in the mapping repository (i.e., the repository maps), the mapping system would not be able to perform its functions.”); *see also* ‘730 Patent at Figure 2 (illustrating use of repository maps). Dr. Jagadish explains that “mapping” is “a term of art and it means the identification of correspondences between data items in two different representations.” *Id.* ¶ 89. He opines that “term ‘repository maps,’ when analyzed in the context of Claim 1 and the specification of the ‘730 patent, obviously refers to the maps in a mapping repository, which in turn is part of the mapping system. The maps in the mapping repository, which are referred to as ‘repository maps’ at times, are essentially rules that govern how data from one data source should be correlated with and/or translated into data from another data source. The specification of the ‘730 patent describes in detail how the repository maps can be imported into or exported from the mapping repository as well as how the repository maps are managed.” *Id.* ¶ 90; *see also* 18:31-41 (defining “repository”).

Given the descriptions of repository maps (as well as the definition of repository and the well-known meaning of mapping), I disagree with Oracle and find that the purpose and role of repository maps are adequately defined and explained in the specification. *See, e.g.*, ‘481 Patent 4:9-10; 4:25-36; 19:5-25.

With respect to “navigational model or schema,” Thought notes that model has been previously defined and schema is readily understood and not disputed. For the particular use of “navigations model,” Thought relies on the definitions of navigational model found in the related ‘481 and ‘600 patents. ‘481 patent at 5:41-45; ‘600 patent at 5:35-39 (“Navigation model” as used herein is a special type of application model that is applied specifically to a description (or other representation) of how objects can relate to each other and what might be the expected behavior when a CDOG is navigated for a certain purpose.”). Dr. Jagadish declares that in light of these definitions, “organizing the repository maps according to a navigation model or schema” means that the “repository maps are organized according to how objects are related to each other and what may be expected when navigating a complex data object graph. Since a user often accesses objects based on their relationships as specified in a navigation model or schema, organizing the repository maps according to the navigation model or schema may help improve

the performance of the mapping system.” *Id.* ¶ 94. Applying the definition of “navigation model” as used in the ‘481 and ‘600 patent (in absence of any argument by Oracle as to why those definitions would not apply similarly in the context of the ‘730 Patent), I conclude that organizing repository maps according to a navigational model or schema is adequately defined for a person skilled in the art and not indefinite.

Therefore, **wherein repository maps are organized according to at least one navigation model or schema** is defined as its component part, using the definitions available in the ‘730, ‘481 and ‘600 Patents.

III. TERMS ORACLE ALLEGES ARE MEANS-PLUS-FUNCTION AND INDEFINITE FOR FAILING TO DISCLOSE CORRESPONDING STRUCTURE

Oracle contends that six terms are indefinite under 35 U.S.C. § 112(f) as means-plus-function claims that lack disclosure of an adequate structure in the patent specifications. I must first determine whether each disputed term is appropriately considered means-plus-function and, if so, whether the required structure is disclosed in the relevant specifications.

A. Legal Standard

As with the § 112(b) challenge discussed above, Oracle bears burden of demonstrating that the claims are means-plus-function and lack the required structure by a preponderance of the evidence. *Apple Inc. v. Motorola, Inc.*, 757 F.3d 1286, 1298 (Fed. Cir. 2014). Whether claim language invokes Section 112(f) is a question of law. *Id.* at 1296. § 112(f) provides:

Element in claim for a combination. An element in a claim for a combination may be expressed as a means or step for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification and equivalents thereof.

“The overall means-plus-function analysis is a two-step process. . . . In the first step, we must determine if the claim limitation is drafted in means-plus-function format. As part of this step, we must construe the claim limitation to decide if it connotes ‘sufficiently definite structure’ to a person of ordinary skill in the art, which requires us to consider the specification (among other evidence). In the second step, if the limitation is in means-plus-function format, we must specifically review the specification for ‘corresponding structure.’ Thus, while these two

1 ‘structure’ inquiries are inherently related, they are distinct.” *Id.* at 1296; *see also id.* at 1296-97
2 (noting that prosecution history and relevant extrinsic evidence can inform the first step).

3 Importantly, “when a claim limitation lacks the term ‘means,’ it creates a rebuttable
4 presumption that Section 112[f] does not apply.” *Id.* at 1297. The presumption is “strong” and
5 “not readily overcome.” *Id.* However, the presumption may be overcome if the claim fails to
6 recite “sufficiently definite structure” or merely recites a “function without reciting sufficient
7 structure for performing that function.” *Id.*

8 As the Federal Circuit recently explained, “[s]tructure’ to a person of ordinary skill in the
9 art of computer-implemented inventions may differ from more traditional, mechanical structure.
10 For example, looking for traditional ‘physical structure’ in a computer software claim is fruitless
11 because software does not contain physical structures. Indeed, the typical physical structure that
12 implements software, a computer, cannot be relied upon to provide sufficiently definite structure
13 for a software claim lacking ‘means.’ Rather, to one of skill in the art, the ‘structure’ of computer
14 software is understood through, for example, an outline of an algorithm, a flowchart, or a specific
15 set of instructions or rules.” *Apple Inc. v. Motorola, Inc.*, at 1298.

16 Moreover, “[a] limitation has sufficient structure when it recites a claim term with a
17 structural definition that is either provided in the specification or generally known in the art.” *Id.*
18 at 1299. “Structure may also be provided by describing the claim limitation’s operation, such as
19 its input, output, or connections. The limitation’s operation is more than just its function; it is how
20 the function is achieved in the context of the invention.” *Id.* Therefore, “if a limitation recites a
21 term with a known structural meaning, or recites either a known or generic term with a sufficient
22 description of its operation, the presumption against means-plus-function claiming remains
23 intact.” *Id.* at 1300. However, “if the claim merely recites a generic nonce word and the
24 remaining claim language, specification, prosecution history, and relevant external evidence
25 provide no further structural description to a person of ordinary skill in the art, then the
26 presumption against means-plus-function claiming is rebutted.” *Id.*

27 **B. Challenged Terms**

28 **1. Computer logic capable of persisting an indicated object or set of objects**

a. Is this means-plus-function language?

As explained in *Apple v. Motorola*, the question is if this term has a known structural meaning or whether the claim language contains a sufficient description of its operation that person skilled in the art would know what to do. Here, “computer logic” is not a known structural meaning; it is simply a process. The claim language does not explain how the logic works, but simply its function: to persist objects.

This term is unlike the term reviewed in *Apple Inc. v. Motorola, Inc.*, where the Federal Circuit reversed the trial court’s conclusion that the term “heuristics” was a means-plus-function description without disclosure of sufficient structure. There, the patent claimed a device with “programs” including:

instructions for applying one or more *heuristics* to the one or more finger contacts to determine a command for the device; and

...

wherein the one or more *heuristics* comprise:

a vertical screen scrolling heuristic for determining that the one or more finger contacts correspond to a one-dimensional vertical screen scrolling command rather than a two-dimensional screen translation command *based on an angle of initial movement of a finger contact with respect to the touch screen display*;

a two-dimensional screen translation heuristic for determining that the one or more finger contacts correspond to the two-dimensional screen translation command rather than the one-dimensional vertical screen scrolling command *based on the angle of initial movement of the finger contact with respect to the touch screen display*; and

a next item heuristic for determining that the one or more finger contacts correspond to a command to transition from displaying a respective item in a set of items to displaying a next item in the set of items.

Id. 757 F.3d at 1295. The Federal Circuit held that the concept of heuristics not only had a known meaning by persons ordinarily skilled in the art¹¹ but it was also described both in the remaining claim language as well as in the specification by its “operation” including its “input, output, and how its output may be achieved.” *Id.* at 1301. Here, the claim language, even when reviewed

¹¹ “[A] person of ordinary skill in the art would understand ‘heuristic’ to mean ‘one or more rules to be applied to data to assist in drawing inferences from that data.’ In this sense, ‘heuristic’ is similar to words that define a class of structures, such as ‘connector,’ ‘circuit,’ and ‘detector,’ and it does not include all means for performing the recited function.” *Apple Inc. v. Motorola, Inc.*, 757 F.3d at 1301.

within the context of the disclosures in the specification, does not explain how the logic at issue operates to perform the function.¹²

Therefore, I find that, even reviewing the specification in order to properly construe this term, “computer logic” invokes means-plus-function claiming. *See, e.g., Visual Networks Operations, Inc. v. Paradyne Corp.*, 2005 WL 1411578 (D. Md. June 15, 2005) (“‘Logic for determining at least one dedicated time slot(s)’ describes only a function, not a structure. Any number of different algorithms, in the form of either computer code or hard-wired circuit logic, could perform the recited function.”); *ABB Automation Inc. v. Schlumberger Res. Mgmt. Svcs., Inc.*, 2003 U.S. Dist. LEXIS 5002 (D.Del. Mar. 27, 2003) (“The court finds that ‘logic’ does not recite sufficient structure to avoid means-plus-function analysis.”); *see also Transperfect Global, Inc. v. MotionPoint Corp.*, C 10-2590 CW, 2013 WL 2299621 (N.D. Cal. May 24, 2013) (“Here, the patents’ claims recite function without reciting any definite structure. The relevant claims do not identify any specific hardware or software for performing the functions listed in each claim. Instead, they refer simply to an ‘apparatus ... comprising: a module for’ performing those functions. These generic references do not recite a sufficiently definite structure.”).¹³

b. Is there sufficient structure disclosed?

The parties agree that the function at issue is “persisting an indicated object or sets of objects.” In its briefing, Thought relied on various portions of the specification to disclose the

¹² This case is also unlike *Fujifilm Corp. v. Motorola Mobility LLC*, 2013 U.S. Dist. LEXIS 165165 (N.D. Cal. Nov. 18, 2013) where I held that “face judgment device” was not subject to § 112, ¶ 6 because when read in light of the specification, the claim language adequately disclosed the structure to perform that function. *Id.* at * 51-52.

¹³ Thought relies on inapposite cases to argue that “logic” when used with other terms provides sufficient structure. In *Apex Inc. v. Raritan Computer, Inc.*, 325 F.3d 1364, 1373 (Fed. Cir. 2003), the Court concluded that “circuit” itself “connotes some structure” and that the term when used “with an appropriate identifier such as ‘interface,’ ‘programming’ and ‘logic,’ certainly identifies some structural meaning to one of ordinary skill in the art.” *Id.* at 1373-74. That case, dealing with physical circuits, does not help Thought here. In *Intel Corp. v. VIA Techs.*, 319 F.3d 1357, 1366 (Fed. Cir. 2003), the Court concluded that “the core logic of a computer” defined there as the “chipset,” “modified to perform [a specific purpose] is the corresponding structure for the functions recited.” Logic there was used in a completely different manner (chipset) than here. Finally, in *TecSec, Inc. v. IBM*, 731 F.3d 1336, 1348 (Fed. Cir. 2013), the Court concluded that “digital logic means to perform” was not mean-plus-function because the term “designates structure to skilled artisans—namely digital circuits that perform Boolean algebra.” The same type of phrase and disclosures are not made in the “computer logic” term at issue here.

structure, including discussions of object implementation through persistence software, which is disclosed in the bottom block of Figure 3 in the ‘600 Patent, 10:25-26. The bottom block of Figure 3, however, simply discloses a “Persistence Library or API for Persisting Objects, Links, Object Data, and/or an Object Model.” This “black box” does not disclose an algorithm or otherwise instructions for accomplishing the function. *See, e.g., Blackboard, Inc. v. Desire2Learn, Inc.*, 574 F.3d 1371, 1383 (Fed. Cir. 2009). (“The ACM is essentially a black box that performs a recited function. But how it does so is left undisclosed.”).

Thought also relies on a preferred embodiment discussed in the Specification that, according to Thought, “disclose[s] extremely specific examples of the algorithm” by disclosing use of Thought’s CocoBase product, including CocoNavigator API. ‘600 Patent 12:19-29; 14:4-17-34; 15:13-20:52. Similarly, at the Claim Construction hearing, Thought argued that the specification discloses a sufficient structure by referring to the Java Data Base Connectivity (JBDC) application programming interface (API).¹⁴ Finally, Thought argues that since “persisting” is a “straightforward process” understood by someone skilled in the art, there is no need to disclose a complex algorithm for accomplishing the function, especially when Thought’s CocoBase product could accomplish it. Reply at 22.

Oracle does not dispute that the JBDC and CocoBase software programs are disclosed in the patents. Oracle does not dispute that these programs are sufficiently “linked” to the claimed function, and Oracle does not dispute (and its expert admits) that CocoBase and JBDC can perform the claimed function.¹⁵ Instead, Oracle argues that software – even software specifically identified in the specification and linked to the function – is not sufficient and an “algorithm” is

¹⁴ Thought did not mention JBDC in its briefs, and its expert, Dr. Jagadish, does not mention JBDC in his declaration. Instead, Dr. Jagadish states only that that a person of ordinary skill in the art would know exactly what to do and how to “persist” an object. Jagadish Resp. Decl. ¶¶ 118-19.

¹⁵ At most Dr. Hosking states that disclosure of the CoCoBase API “by definition does not reveal anything about the algorithms that implement that interface.” Hosking Decl. ¶ 252; Hosking Sur-Reply Declaration (Docket No. 87-1) ¶¶ 8-9 (“The citation to column 12 merely refers to commercially available APIs, and states that those APIs, in conjunction with a commercial embodiment of the mapping tool, work together to ‘manipulate complex objects.’ There is, however, no description of the actual interactions between the API and the mapping tool, or any description of the process or steps undertaken to accomplish persistence.”).

required. Sur-Reply 8-9. However, the Federal Circuit has recognized that when a specification discloses a specific type of software that is “linked” to (in other words can be used to perform) the function, that is sufficient. *See Med. Instrumentation & Diagnostics Corp. v. Elekta AB*, 344 F.3d 1205, 1214 (Fed. Cir. 2003) (“There was no need for a disclosure of specific circuitry in that case, just as here there would be no need for a disclosure of the specific program code if software were linked to the converting function and one skilled in the art would know the kind of program to use.”); *see also id.* at 1212 (“The correct inquiry is to look at the *disclosure* of the patent and determine if one of skill in the art would have understood that *disclosure* to encompass software for digital-to-digital conversion and been able to implement such a program, not simply whether one of skill in the art would have been able to write such a software program.” (emphasis in original)).

Oracle relies on *Noah Sys. Inc. v. Intuit Inc.*, 675 F.3d 1302 (Fed. Cir. 2012). However, that case did not reject the idea that software products specifically identified in the specification could not satisfy the disclosed structure requirement. Instead, in that case, the patentee relied on the specification’s mention of “off the shelf software” from a part of the specification dealing with a *different* claim and then asserted “individuals of ordinary skill in the art would understand how to accomplish the function described with the assistance of such off the shelf software.” *Id.* at 1317. The Federal Circuit rejected the patentee’s arguments because the disclosure itself must identify the method for performing the function “whether or not a skilled artisan might otherwise be able to glean such a method from other sources or from his own understanding,” and concluded that the patentee’s “efforts to find structure in the portion of a specification linked to a different claim element or in the common ken of a skilled computer artisan does not allow it to ‘avoid providing [the] specificity as to structure’ required by § 112 ¶ 6.” *Id.* The other cases relied on by Oracle are also inapposite as the claim language, at most, identified “software” or a “computer” to perform the requisite function. *See Fujifilm Corp. v. Motorola Mobility LLC*, 12-CV-03587-WHO, 2013 WL 6185254 (N.D. Cal. Nov. 18, 2013 (finding insufficient structure where reference was to a “CPU [] ‘programmed to calculate’ its function, without adequately disclosing the structure or method for performing such a calculation or such programming, effectively leaving

the CPU a general purpose computer.”); *Encyclopaedia Britannica, Inc. v. Alpine Electronics of Am., Inc.*, 2008 WL 7328271 (W.D. Tex. Sept. 30, 2008) *aff’d sub nom. Encyclopaedia Britannica, Inc. v. Alpine Electronics, Inc.*, 355 F. App’x 389 (Fed. Cir. 2009) (“A general reference to ‘software’ is not sufficient to disclose structure,” rejecting claim that software not disclosed in the specification was an adequately disclosed algorithm); *Touchcom, Inc. v. Dresser, Inc.*, 427 F. Supp. 2d 730, 736 (E.D. Tex. 2005) (rejecting patentee’s argument that a branded device mentioned in the patent in conjunction with its publicly available software drivers could perform the recited function because “[t]hese drivers are not sufficiently disclosed and referenced in the specification to supply the corresponding structure for controlling the input means.”).

At the Claim Construction hearing, Thought relied exclusively on the disclosure of JBDC, arguing that Dr. Hosking himself testified in his deposition that JBDC alone is a sufficient structure to persist an object. Hosking Depo. at 18:03:04. In his Supplemental Declaration – submitted at the Claim Construction hearing – Dr. Hosking admits that JBDC can be “used” to persist objects, but that a programmer would still need to develop and sequence the specific commands that JBDC uses to transmit to the database. Hosking Supplemental Declaration (Docket No. 105), ¶¶ 12-13. Therefore, according to Dr. Hosking, different programmers can “use” JBDC in different ways to accomplish the persisting function and can also use the JBDC without knowing how a particular JBDC driver (created by database developers to provide a JBDC interface for their databases) is implemented. *Id.* ¶¶ 13-14. Dr. Hosking does not explain the *significance* of the fact that programmers could use different sequences of commands to use JBDC to persist data, and he does not otherwise challenge Thought’s position that a person skilled in the art could use the disclosed JBDC to “persist” data as required by the claim language. Instead, based on his belief that an “algorithm” is required by Federal Circuit precedent to satisfy means-plus-function structure, he simply notes that the “bare reference” to “use JBDC” does not disclose an algorithm for implementing the data persisting function. *Id.* ¶ 15.

Relying on Dr. Jagadish’s undisputed position that a person skilled in the art could use the disclosed CocoBase program and API, as well as Dr. Hosking’s agreement that the disclosed JBDC could also be used to persist data as required by the claim, and in light of persuasive Federal

Circuit authority allowing the “algorithm” requirement to be satisfied by a disclosed piece of software, I find that that “computer logic” has been sufficiently described.

2. Logic for assessing each schema corresponding to a different data source accessible to the mapping system

a. Is this means-plus-function language?

As with “computer logic,” “logic for assessing” is not a known structural meaning; it is a process. Despite the lack of the term “means,” construing the claim language in light of the specification, I find the term “logic for assessing” invokes means-plus-function claiming.

b. Is there sufficient structure disclosed?

The parties agree that the function at issue is logic for “accessing each schema corresponding to a different data source accessible to the mapping system.”¹⁶ Thought argues that CocoBase and JBDC are disclosed in the specification as the means to perform the function of creating logic for accessing the schemas.¹⁷ Dr. Jagadish addresses the CocoBase products (Declaration of Hosagrahar Jagadish (Docket No. 76-1) ¶¶ 138-40) but does not address JBDC products specifically. He argues that searching for and accessing databases with differing schemas are “fundamental tasks” in the field, algorithms to do so are taught in basic computer science classes, and those skilled in the art can readily accomplish it. Jagadish Resp. Decl. ¶ 122.

Oracle and its expert do not dispute that disclosure of CocoBase and JBDC is made repeatedly in the specification for the ‘730 Patent and “linked” to the function to be performed. Instead, Oracle argues that the disclosure of those software programs is insufficient because there still is no “algorithm” provided that explains in the text of the specification how the steps for accessing the schemas of the different databases take place. Oracle Response Brief at 36. Dr. Hosking states that the portion of the specification discussing CocoBase does not mention “accessing schemas,” Hosking Decl. ¶ 266, but acknowledges that JBDC could be used to access

¹⁶ A “‘database schema’ or ‘structural schema’ represents the tables in a relational database and their relationships.” Hosking Decl. ¶ 66.

¹⁷ Thought did not rely on JBDC in its briefing, but only at the Claim Construction Hearing. In its briefing, Thought relied for structure on disclosures in the ‘197 Patent and ‘730 Patent specifications describing the “object schema manager” disclosed in the ‘197 Patent and “equivalents” thereof in the ‘730 Patent. Thought also relied on the disclosure in the specification of CocoBase. Thought Brief at 35; Reply at 22-23.

1 “each schema corresponding to a different data source accessible to the mapping system.”

2 Hosking Supp. Decl. ¶ 10. He argues, as above, that even though JDBC could be used to perform
3 the accessing at issue, because there might be different command sequences to do so and because
4 “use of JDBC” indicated in the specification does not disclose how to use the program to perform
5 the access or how JDBC operates to perform the accessing, it cannot satisfy the algorithm
6 requirement. *Id.* ¶¶ 12-15.

7 Similar to the prior claim, Dr. Hosking does not explain the *significance* of the fact that
8 programmers could use different sequences of commands to use JBDC to access the schemas, and
9 he does not otherwise challenge Thought’s position that a person skilled in the art readily knows
10 how to perform this basic concept, and could do so using JBDC. Relying on the undisputed fact
11 that use of CocoBase and JBDC were disclosed in the specification and adequately linked to
12 accessing the database schemas, I find that the structure of “logic for assessing” is adequately
13 disclosed.

14 **3. Logic for creating or maintaining all or a portion of such data in at least one**
15 **second data source having the same or a different data structural schema**

16 **a. Is this means-plus-function language?**

17 The logic at issue in this term, as an initial matter, is far more complex than simply
18 “persisting” or “assessing.” The logic here must perform complex functions – maintaining all or
19 simply a portion of data, in a data source having the same or a different schema. The claim
20 language, even when considered with the remaining claim language and the specification, does not
21 describe sufficient *operation* – e.g. how the logic achieves its end – to avoid treatment as a mean-
22 plus-function term. I find, therefore, that this term invokes mean-plus-function claiming.

23 **b. Is there sufficient structure disclosed?**

24 The parties agree that the function here is for “creating or maintaining all or a portion of
25 such data in at least one second data source having the same or a different data structural schema.”
26 Thought contends that the structure is disclosed by references to CocoBase, specific code
27 examples that implement the claim, as well as reference to a Java application. Thought Opening
28 Br. at 36; Reply at 23; *see also* ‘730 Patent 32:44-66 (disclosing Thought.CocoBase.SynchPlugin);

1 Jagadish Decl. ¶ 144.¹⁸

2 In response, Dr. Hosking argues that disclosures based on Java are insufficient because the
3 specification fails to disclose how a Java application could be created or how it would access the
4 relevant data. Hosking Decl. ¶ 286. He also argues that the CocoBase reference and code
5 examples are insufficient because they do not encompass how the whole system covered by the
6 ‘730 Patent. *See, e.g.*, Hosking Decl. ¶ 277 (no elaboration on how data is actually transmitted or
7 how mismatches are handled and resolved) ¶ 278 (fails to disclose how the second database is
8 identified and connected to). However, as Dr. Jagadish points out, Java was a well-known
9 program in 2003, and there is no dispute that Java could perform the “creating and maintaining”
10 function. Jagadish Resp. Decl. ¶ 125. As to the functions Dr. Hosking claims are not explicitly
11 addressed in the specification, Dr. Jagadish argues those functions go beyond the “creating and
12 maintaining” feature, which is all that needs to be disclosed and is disclosed in the specification
13 with respect to this claim. *Id.* ¶ 126.

14 At the Claim Construction hearing, Thought again relied exclusively on JBDC because
15 JBDC is disclosed in the specification and at his deposition Dr. Hosking admitted that JBDC could
16 perform “creating and maintaining” feature. Hosking Supp. Decl. ¶ 15. Dr. Hosking’s response in
17 his Supplemental Declaration submitted at the hearing was the same; because there might be
18 different command sequences to do so and because “use of JDBC” indicated in the specification
19 does not disclose how to use the program to perform the “creating and maintaining” function or
20 how JDBC operates to perform the “creating and maintaining,” it cannot satisfy the algorithm
21 requirement. *Id.* ¶¶ 12-15.

22 Similar to the prior claims, Dr. Hosking does not explain the *significance* of the fact that
23 programmers could use different sequences of commands to use JBDC to “create and maintain” as

25 ¹⁸ In particular, with respect to the “creating or maintaining” limitation, Dr. Jagadish relies on the
26 ‘730 Patent at 24:40-58, disclosing that the system “may include a means for transferring the data
27 from a first data source to a second data source. In one embodiment this system includes a simple
28 computer program in a computer language such as Java, which reads the data from the first
database using a repository map that may optionally be cached in the computer memory of the
system and then stores the accessed data to a second database using the same or a different
repository map.” *Id.* ¶ 144.

contemplated, and he does not otherwise challenge Thought's position that a person skilled in the art readily knows how to perform this function, and could do so using Java, CocoBase or JBDC. Relying on the undisputed fact that use of Java, CocoBase, and JBDC was disclosed in the specification and adequately linked to creating and maintaining data in a second data source having a different structure, I find that the structure of "creating and maintaining" is adequately disclosed.

4. A computer programming sub-routine or sub-module for comparing

a. Is this means-plus-function language?

A sub-routine or sub-module, is not a known structural meaning. Nor is the "for comparing" paired with a sufficient description of its operation. The challenged limitation reads in more detail:

a computer programming sub-routine or sub-module for *comparing* a copy of a CDOG, CDOG model, or a representation definition of either the CDOG or CDOG model, to an original stored version of the CDOG, CDOG model, or an original stored representation definition for the CDOG or CDOG model, and for ***updating*** the original to incorporate any changes to a CDOG or a representation definition that are made by the user or by a software program module.

Claim 7, '481 Patent 39:5-14 (emphasis added). This language states the purposes of the comparing and updating but does not, even when read with the remaining claim language and the specification, disclose how the routine or module *operates* to avoid treatment as a mean-plus-function term. I find, therefore, that this term invokes mean-plus-function claiming.

b. Is there sufficient structure disclosed?

Thought argues that the "comparing" function refers to a sub-component of an API, which is disclosed as CocoNavigator API in the specification, and that the specification also provides an example of using Java method "equals" to perform the comparison. Jagadish Decl. ¶ 149.¹⁹ Oracle responds that the API is simply an interface and there is no evidence that it can perform the comparing function, and that the Java equals method cannot disclose the comparing function

¹⁹ At the Claim Construction hearing, Thought relied exclusively on the disclosure of the Java equals method as disclosing the required structure.

because it must be adapted to perform the operation. *See, e.g.*, Hosking Decl. ¶ 291 (“It is my opinion that the Java equals method could not be used, without more, to compare CDOGs or CDOG models, as claimed. Again, the specification does not assert that Java equals is used this way—it limits the use of Java equals to object comparison, not object graph or object graph model comparison.”). Oracle also argues that *nothing* is pointed to by Thought in the specification that could perform the updating function, and that in his deposition Dr. Jagadish admitted that the updating step the function could not be performed by Java equals. Jagadish Depo. at 122:15-22.

As to the Java equals disclosure, I find Dr. Jagadish’s position more persuasive. Dr. Jagadish explains that Dr. Hosking’s overstates the complexity of the specific comparison required by the claim – comparing CDOG and CDOG models – and that Java equals can perform that basic comparison without significant modification. Jagadish Resp. Decl. ¶ 129 (distinguishing the situation where “the objects to be compared include external links to other objects, or have type differences” that might require modification or enhancement of Java equals method, to the type of comparison at issue; “we have two database schemas, represented as graphs. It is straightforward to say whether they are equal, and to identify differences between them to the extent they are not, and the Java method equals is adequate to perform this computer operation.”).

Thought has totally failed to address what the disclosed structure is for the updating function that is part of the claim language to be construed. The issue was raised by Oracle in its briefs (*see* Oracle Resp. Br. at 38), but never responded to by Thought in its briefing or by Dr. Jagadish. Because the Java equals method cannot perform the whole of the claimed function (comparing and updating), sufficient structure has not been disclosed and this claim is indefinite.

5. A feature providing an interface for editing a CDOG model or for editing a definition or other representation thereof

a. Is this means-plus-function language?

The “feature” here is likewise not a known structural meaning. Nor does a feature that can edit a CDOG model or definition or representation thereof, even when read with the remaining claim language and the specification, disclose the operation of the feature. It is instead simply a description of the interface’s multiple functions. I find, therefore, that this term invokes mean-

plus-function claiming.

b. Is there sufficient structure disclosed?

The parties agree that the function is “providing an interface for editing a CDOG model or for editing a definition or other representation thereof.” Thought argues the structure is disclosed in the specification by: (i) identifying the “interface” as “an editing interface for editing the CDOG model, or has an editable input or source, such as a file, that can be modified to implement changes to the complex data object model.” ‘481 Patent 6:37-41; (ii) the specification described a preferred embodiment where this interface is a “point and click graphical user interface” familiar to Java developers and used in the CocoBase product, ‘481 Patent 6:42-43; and (iii) the ‘481 specification identifies CocoNavigator API as an interface can allows users to manipulate the CDOG model. ‘481 Patent 9:10-13; Jagadish Decl. at ¶¶ 152-153. Thought also argues that code, incorporated into the ‘481 Patent (36:56-62), likewise disclosed the readily known, used, and available NavDemo.java that could be used to build the graphical user interface for editing the CDOG model. Thought Reply at 24; *see also* Jagadish Resp. Decl. ¶¶ 132-33.

Oracle does not respond specifically to Thought’s argument to dispute that the specification’s express reference to a graphical user interface, when combined with disclosed use of CocoNavigator API or the NavDemo.java code, would be sufficient structure to disclose the method for the editing at issue. Oracle ignores this argument in its Sur-Reply and Dr. Hosking does not address it. At the Claim Construction hearing Oracle relied on an edited portion of Dr. Jagadish’s deposition, where he affirmed that in his declaration he cited to code that would “inform one skilled in the art how to build a graphical user interface to edit a CDOG model,” but admitted that the codes does not disclose an “algorithm for this specific function explicitly culled out . . .” Jagadish Depo. at 136:20-137:6. However, Dr. Jagadish’s point remains that one skilled in the art could readily use the admittedly disclosed interface solutions and underlying code to implement an interface for editing a CDOG model or definition or representation thereof. *See, e.g., Typhoon Touch Techs., Inc. v. Dell, Inc.*, 659 F.3d 1376, 1385 (Fed. Cir. 2011) (“the patent need only disclose sufficient structure for a person of skill in the field to provide an operative software program for the specified function.”). I find that the structure of “an interface for editing”

is adequately disclosed.

6. A feature . . . providing an interface for an editable input or source, such as a file, that can be modified to implement changes to a CDO or its relationships of a CDOG model

a. Is this means-plus-function language?

As above, this feature is not a known structure. Nor does the claim language, even when read in conjunction with the remaining language and the specification, disclose the operation of the feature. The claim language simply describes this interface's multiple functions. I find, therefore, that this term invokes mean-plus-function claiming.

b. Is there sufficient structure disclosed?

The parties agree that the function is an interface for editing input or source files to implement changes to a CDO or the relationships of a CDOG model. Thought argues that the specification clearly links editable source files as XML, XMI, and UML – which are data source files – to database files, database mapping definition files, and CDOG definition files. '481 Patent 9:16-30. And that a specific example of a CDOG API (which can complete this function) is provided as CocoNavigator API. *Id.* at 11:7-13:60. Dr. Jagadish explains that “XML, XMI, and UML, are straightforward and numerous commercially available and freeware tools are readily available to edit the content of each type of file and would be readily known to those of skill in the art as of the effective filing date of the '481 patent. The identification of the file types represents a sufficiently definite technique for editing the content in view of the ready availability and ease of use of the editing tools.” Jagadish Resp. Decl. ¶ 136.

Oracle responds that the specification never explains how to actually do the XML/XMI/UML editing function, and Dr. Hosking argues that the specification fails to disclose how to provide an interface for the XML/XMI/UML files or how the edits are actually accomplished. Hosking Resp. Decl. ¶ 309. But Dr. Jagadish's point that a person skilled in the art would readily be able to edit and implement changes to those files using the disclosed editing tools stands un rebutted. That the “inner workings” of the readily available editing tools are not disclosed, does not defeat the sufficiency of the disclosure considering that a person skilled in the art is familiar would be familiar with those inner workings and could readily put those tools to use

to perform the function at issue. I find that the structure of “interface for an editable input or source” is adequately disclosed.

CONCLUSION

The claims, therefore, are construed as follows:

Object as used the ‘197 Patent is defined as: **instance of a class.**

Object as used in the ‘600 and ‘862 Patents is defined as: **a general term referring to a logic unit in a computer application or a computer software program, that may be used interchangeably with the term “class” as a template or as an instance depending on the context.**

Object as used in the ‘730 Patent is defined as: **an organized set of encapsulated programming code designed to act on itself at the request of some external system, which system may pass in some additional information to the object in that request.**

Object schema as used in the ‘197 Patent is defined as: **object-oriented view of one or more data store schemas.**

Packing as used in the ‘197 Patent is defined as: **placing data into a form suitable for communication between software components.**

Unpacking as used in the ‘197 Patent is defined as: **retrieving data from a packed data structure.**

Interface as used in the ‘197 Patent is defined as: **part of an adapter abstraction layer that facilitates interaction between software components.**

Adapter as used in the ‘197 Patent is defined as: **runtime interchangeable part of an adapter abstraction layer that enables incompatible software components to work together.**

Adapter abstraction layer as used in the ‘197 Patent is defined as: **software components with interfaces provided by adapters.**

Fully synchronized caching system as used in the ‘956 Patent is defined as: **a caching system in which any changes to the cache will automatically be synchronized with the primary data source or record upon transaction completion.**

Extensive knowledge as used in the ‘956 Patent is defined as: **knowledge of the schema**

1 **and query language sufficient to save or retrieve data.**

2 **To provide . . . to a system user who is** as used in the ‘730 and ‘956 Patents is defined as:
3 **to provide a system user the ability to make data changes related to a particular object as**
4 **either local or global changes on the computer system.**

5 **Model** as used in the ‘600 and ‘862 Patents is defined as: **logical or abstract**
6 **representation or description.**

7 **Changes to a CDO or its relationship of a CDOG model** as used in the ‘481 Patent is
8 defined as **changes to (1) a CDO, or (2) relationships of the CDO, the CDO and its**
9 **relationships being part of a CDOG model.**

10 **The information of (a)** as used in the ‘481 Patent is defined as: **information read from a**
11 **source programming object logic model or a database definition file in a format selected**
12 **from the group consisting of a UML data file, a XML data file, and a XML file ((a) refers to**
13 **limitation 12-(a)).**

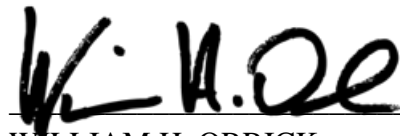
14 **Wherein repository maps are organized according to at least one navigation model or**
15 **schema** as used in the ‘730 Patent is defined as its component part, using the definitions available
16 in the ‘730, ‘481 and ‘600 Patents.

17 I also reject Oracle’s indefinite challenges, except for “A computer programming sub-
18 routine or sub-module for comparing . . . and updating” which I find is indefinite for failure to
19 adequately disclose sufficient structure to perform the claimed updating function.

20 A Case Management Conference will be held at 2:00 p.m. on November 19, 2014 in
21 Courtroom 2. The parties shall file a Joint Case Management Statement on or before November
22 13, 2014 that proposes a case schedule through trial (jointly if possible, separately if not).

23 **IT IS SO ORDERED.**

24 Dated: October 22, 2014

25 

26 WILLIAM H. ORRICK
27 United States District Judge
28